Projektteams Lexer / Parser / Zwischencode
Ideenbesprechung vom 28.04.

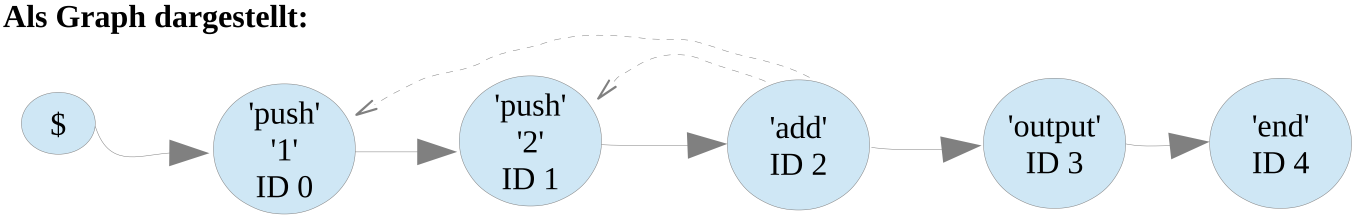
→ wir hatten uns überlegt, ein geparstes Rail-Programm in eine Graph-Struktur umzuwandeln

Beispielprogramm in Rail:

```
$ –
    \
     \
       ----- [1] ------- [2] ------- a ----- o ----- #
```

→ liest die Zahlen *1* und *2*, addiert sie und gibt das Ergebnis aus

Als Graph dargestellt:



Als Tabelle dargestellt:

ID	Command	Adj (Nachfolgeknoten)
0	push('1')	1
1	push('2')	2
2	add()	3
3	output()	4
4	end()	-

→ hier noch ein paar Besonderheiten, die auch für die späteren Meilensteine relevant sind

Reverse-Operator:

[…] -- *a* -- @

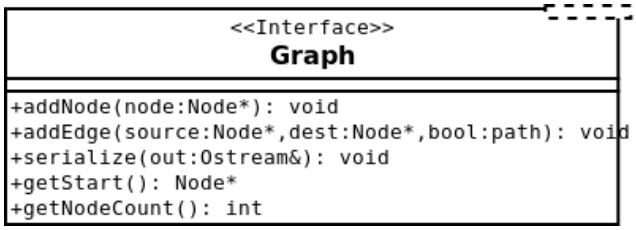
→ für *a* werden zwei verschiedene Knoten angelegt (einmal auf dem Hin- und einmal auf dem Rückweg)

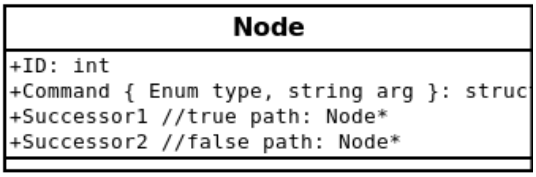
Schleifen behandeln:

```
[…] ---- [a b c] -----
                \
    ------------------
   /                  \
   \___  a  _________/
```

→ *a* wird **einmal** in einem Knoten gespeichert
→ der Parser darf nicht endlos über *a* laufen und neue Knoten anlegen

Konzept für die C++-Klassen:





Aufgabenverteilung für den ersten Meilenstein (15.05.):

Was?	Wer?
Graph & Node implementieren	Hanin
Serialisieren d. Graphen	Miro
Railprogramm → Graph	Leon, Stefan
Deserialisieren	Chris
Diverse Namen (Funktionen)	Sandra
Rail-Testprogramme	Tomasz
Codequalität bewerten	Houssem